

Exp – 2

REGNO-RA1911026010119

NAME-SANKALP JAIN

Graph Coloring problem

Lab-2

BA1911026 01/01/19

Sankalp Jain

aim \rightarrow To solve & perform Vertex & edge coloring

Algorithm-

(1) Vertex coloring

- \rightarrow Add list of colors.
- \rightarrow Then give list of graph edges & nodes and create function to build graph from edges.
- \rightarrow convert edges to undirected graph.
- \rightarrow Create function to assign colors to vertex.
- \rightarrow Check for 1st color and assign all others.
- \rightarrow Loop Next vertex and color it with least numbered color that has been colored on any vertices adjacent to it.
- \rightarrow Print.

(2) Graph edge coloring

- \rightarrow Define graph by Number of Vertices & edge connected by each other. then create function to determine edge colors.
- \rightarrow Traverse all edges one by one & check them.
- \rightarrow If color vertex found to be negative then assign color and increment color i.e. change color for Next.
- \rightarrow Add it to the set.
- \rightarrow Repeat until all edges of graph are colored.

Code:

VERTEX COLORING

```
class Graph:
```

```
    def __init__(self, edges, n):
```

```
        self.adjList = [[] for _ in range(n)]
```

```
        # add edges to the undirected graph
```

```
        for (src, dest) in edges:
```

```
            self.adjList[src].append(dest)
```

```
            self.adjList[dest].append(src)
```

```
def colorGraph(graph, n):
```

```
    result = {}
```

```
    for u in range(n):
```

```
    assigned = set([result.get(i) for i in graph.adjList[u] if i in
result])
```

```
    color = 1
```

```
    for c in assigned:
```

```
        if color != c:
```

```
            break
```

```
    color = color + 1
```

```
result[u] = color
```

```
for v in range(n):
```

```
    print(f'Color assigned to vertex {v} is {colors[result[v]]}')
```

```
if __name__ == '__main__':
```

```
    colors = ['', 'GREEN', 'ORANGE', 'RED', 'CYAN', 'PURPLE',
'PINK',
```

```
    'PINK', 'BLACK', 'GREEN', 'CYAN', 'BLUE']
```

```
    edges = [(0, 1), (0, 2), (1, 5), (2, 5), (2, 4), (2, 3), (0, 3), (1, 4)]
```

```
n = 6
```

```
# build a graph from the given edges
```

```
graph = Graph(edges, n)
```

```
colorGraph(graph, n)
```

EDGE COLORING

```
from queue import Queue
```

```
def colorEdges(ptr, gra, edgeColors, isVisited):
```

```
    q=Queue()
```

```
    c = 0
```

```
    colored=set()
```

```
    if (isVisited[ptr]):
```

```
        return
```

```
    # Mark the current node visited
```

```
    isVisited[ptr] = True
```

```
for i in range(len(gra[ptr])) :
```

```
    if (edgeColors[gra[ptr][i][1]] != -1):
```

```
        colored.add(edgeColors[gra[ptr][i][1]])
```

```
for i in range(len(gra[ptr])) :
```

```
    if not isVisited[gra[ptr][i][0]]:
```

```
        q.put(gra[ptr][i][0])
```

```
    if (edgeColors[gra[ptr][i][1]] == -1) :
```

```
        while c in colored:
```

```
            # increment the color
```

```
            c+=1
```

```
# copy it in the vector  
edgeColors[gra[ptr][i][1]] = c
```

```
# then add it to the set  
colored.add(c)  
c+=1
```

```
while not q.empty() :  
    temp = q.get()
```

```
    colorEdges(temp, gra, edgeColors, isVisited)
```

```
return
```

```
# Driver Function
```

```
if __name__=='__main__':  
    empty=set()  
  
    gra=[]  
  
    edgeColors=[]  
  
    isVisited=[False]*100000  
  
    # Enter the Number of Vertices  
    # and the number of edges  
    ver = 4  
    edge = 4  
  
    gra=[[[] for _ in range(ver)]  
    edgeColors=[-1]*edge  
  
    gra[0].append((1, 0))  
    gra[1].append((0, 0))
```



```
gra[1].append((2, 0))
```

```
gra[2].append((1, 1))
```

```
gra[2].append((1, 2))
```

```
gra[3].append((2, 2))
```

```
gra[0].append((0, 3))
```

```
gra[3].append((0, 3))
```

```
colorEdges(0, gra, edgeColors, isVisited)
```

```
# printing all the edge colors
```

```
for i in range(edge):
```

```
    print("Edge {} is of color {}".format(i + 1, edgeColors[i]
+ 1))
```

Output:

Vertex coloring:

```
1 class Graph:
2     def __init__(self, edges, n):
3         self.adjlist = [[] for _ in range(n)]
4
5         # add edges to the undirected graph
6         for (src, dest) in edges:
7             self.adjlist[src].append(dest)
8             self.adjlist[dest].append(src)
9
10    # function to assign colors to vertices of a graph
11    def colorGraph(self, n):
12        # some track of the color assigned to each vertex
13        result = 0
14
15        # assign a color to vertex one by one
16        for u in range(n):
17            # check colors of adjacent vertices of 'u' and store them in a set
18            assigned = set([result.get(i) for i in graph.adjlist[u] if i in result])
19
20            # check for the first free color
21            while True:
22                result[u] = result.get(u, 0)
23                if result[u] not in assigned:
24                    break
25                result[u] += 1
```

Color assigned to vertex 0 is GREEN
Color assigned to vertex 1 is ORANGE
Color assigned to vertex 2 is ORANGE
Color assigned to vertex 3 is RED
Color assigned to vertex 4 is GREEN
Color assigned to vertex 5 is GREEN

Edge coloring:

```
1 class Graph:
2     def __init__(self, edges, n):
3         self.adjlist = [[] for _ in range(n)]
4
5         # add edges to the undirected graph
6         for (src, dest) in edges:
7             self.adjlist[src].append(dest)
8             self.adjlist[dest].append(src)
9
10    # function to assign colors to edges of a graph
11    def colorEdges(self, n):
12        # some track of the color assigned to each edge
13        result = 0
14
15        # assign a color to edge one by one
16        for u in range(n):
17            # check colors of adjacent vertices of 'u' and store them in a set
18            assigned = set([result.get(i) for i in graph.adjlist[u] if i in result])
19
20            # check for the first free color
21            while True:
22                result[u] = result.get(u, 0)
23                if result[u] not in assigned:
24                    break
25                result[u] += 1
```

Edge 1 is of color 1
Edge 2 is of color 1
Edge 3 is of color 2
Edge 4 is of color 2

Result:

Hence Graph Coloring problem for both vertex coloring and Edge coloring were studied and solved.